Zapier TEST

I didn’t know how to use the tool, I checked a little bit so I would have needed to do the following

Conditional on subject

Here are the codes for the 2 other problems

Weather  
  
import requests

import pandas as pd

# Define the API endpoint

endpoint = "https://open-meteo.com/en/docs/historical-weather-api"

# Set the parameters

params = {

"latitude": 34.685641569710285,

"longitude": 135.5069510274213,

"start\_date": "2024-05-01",

"end\_date": "2024-05-05",

"hourly": "temperature\_2m",

"timezone": "Asia/Tokyo"

}

# Make the API request

response = requests.get(endpoint, params=params)

# Check if the request was successful

if response.status\_code == 200:

data = response.json()

else:

print(f"Error: {response.status\_code}")

data = None

if data:

# Extract time and temperature data

times = data["hourly"]["time"]

temperatures = data["hourly"]["temperature\_2m"]

# Create a DataFrame

df = pd.DataFrame({

"time": times,

"temperature\_2m": temperatures

})

# Save to CSV

df.to\_csv("temperature\_data.csv", index=False)

print("Data saved to temperature\_data.csv")

import pandas as pd

# Load the temperature data from CSV

df = pd.read\_csv("temperature\_data.csv")

# Convert time column to datetime format

df["time"] = pd.to\_datetime(df["time"])

# Extract date from timestamp

df["date"] = df["time"].dt.date

# Compute daily statistics

daily\_summary = df.groupby("date")["temperature\_2m"].agg(

max\_temp="max",

min\_temp="min",

avg\_temp="mean"

).reset\_index()

# Save to CSV

daily\_summary.to\_csv("daily\_temperature\_summary.csv", index=False)

print("Daily summary saved to daily\_temperature\_summary.csv")

import pandas as pd

import matplotlib.pyplot as plt

# Load the daily temperature summary

df = pd.read\_csv("daily\_temperature\_summary.csv")

# Convert date column to datetime format for better plotting

df["date"] = pd.to\_datetime(df["date"])

# Plot the data

plt.figure(figsize=(10, 5))

plt.plot(df["date"], df["max\_temp"], marker="o", linestyle="-", label="Max Temperature", color="red")

plt.plot(df["date"], df["min\_temp"], marker="o", linestyle="-", label="Min Temperature", color="blue")

plt.plot(df["date"], df["avg\_temp"], marker="o", linestyle="-", label="Avg Temperature", color="green")

# Formatting the plot

plt.xlabel("Date")

plt.ylabel("Temperature (°C)")

plt.title("Daily Temperature Trends (Max, Min, Avg)")

plt.legend()

plt.grid(True)

# Rotate x-axis labels for better visibility

plt.xticks(rotation=45)

# Save the plot as a JPG file

plt.savefig("temperature\_trends.jpg", dpi=300, bbox\_inches="tight")

plt.show()

print("Plot saved as temperature\_trends.jpg")

Currency  
  
import requests

import pandas as pd

import time

# Define base URL and date range

base\_url = "https://www.xe.com/currencytables/?from=JPY&date={date}#table-section"

dates = ["2024-05-01", "2024-05-02", "2024-05-03", "2024-05-04", "2024-05-05"]

# List to store all data

all\_data = []

# Loop through each date and scrape data

for date in dates:

url = base\_url.format(date=date)

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"

}

response = requests.get(url, headers=headers)

if response.status\_code != 200:

print(f"Failed to retrieve data for {date}")

continue

# Read HTML tables using pandas

tables = pd.read\_html(response.text)

if not tables:

print(f"No table found for {date}")

continue

# Extract exchange rate table

df = tables[0]

# Rename columns for clarity

df.columns = ["Currency Code", "Currency Name", "Units per JPY", "JPY per Unit"]

# Add date column

df["Date"] = date

# Store data

all\_data.append(df)

# Respectful scraping: Wait before the next request

time.sleep(2)

# Combine all data into a single DataFrame

final\_df = pd.concat(all\_data, ignore\_index=True)

# Create a dictionary to store data by currency code

currency\_data = {}

# Split data by currency code and save to a dictionary

for currency\_code in final\_df["Currency Code"].unique():

currency\_data[currency\_code] = final\_df[final\_df["Currency Code"] == currency\_code]

# Save the data to an Excel file with each currency in a separate sheet

with pd.ExcelWriter("currency\_exchange\_rates.xlsx") as writer:

for currency\_code, data in currency\_data.items():

data.to\_excel(writer, sheet\_name=currency\_code, index=False)

print("Data has been saved to 'currency\_exchange\_rates.xlsx'")